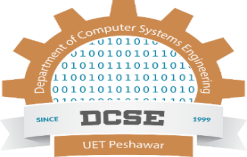
****

**University of engineering & technology Peshawar**

**Object oriented programming-lab**

**Lab report no#02**

**Fall 2020**

**Submitted by: Ashfaq Ahmad**

**Section: B**

**Reg No: 19PWCSE1795**

**Semester: 3nd**

**“On my honor, as a student of University of Engineering and Technology Peshawar, I have neither given nor received unauthorized assistance on this academic work”**

Student signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted to:**

**Eng: Sumayyea salahuddin**

**Department Of Computer System Engineering**

**Task no 1:**

**Reuse Complex class given in section 2.3.2 (C++), section 2.3.5 (Java), and section 2.3.6 (Python) to perform arithmetic with complex numbers. Note that addition and printing is already done in given sections. Add the following public methods to perform complex subtraction and multiplication as well:**

**a) Input: Write class function input() to take complex number real and imaginary parts from user on runtime. Note: input takes no arguments and returns nothing.**

**b) Subtract two Complex numbers: Write class function subCom() taking two complex objects c1 and c2. Difference is computed as following: the real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand. Note: return type of subCom is void.**

**c) Multiply two Complex numbers: Write class function mulCom() taking two complex objects c1 and c2. Product is computed as following: Suppose you are trying to compute the product of two complex numbers a + bi and c + di. The real part will be ac – bd, while the imaginary part will be ad+ bc. Note: return type of subCom is void.**

**Source code:**

#include<iostream>

using namespace std;

class complex

{

private:

float real,imag;

public:

complex()

{

real=0;

imag=0;

}

complex(float r,floati)

{

real=r;

imag=i;

}

// Using Friend Function:

void sum(complex c1,complex c2)

{

real = c1.real + c2.real;

imag = c1.imag + c2.imag;

}

void sub(complex c1,complex c2)

{

real = c1.real - c2.real;

imag = c1.imag - c2.imag;

}

void multiply(complex c1,complex c2)

{

real=c1.real\*c2.real - c1.imag\*c2.imag;

imag=c1.real\*c2.imag + c1.imag\*c2.real;

}

void show()

{

cout<<endl<<real<<" , "<<imag<<"i"<<endl;

}

};

int main()

{

float real,imag;

cout<<"Enter First complex Number: "<<endl;

cout<<"Real Part: ";

cin>>real;

cout<<"Imagenary Part: ";

cin>>imag;

complex c1(real,imag);

cout<<endl<<"Enter Second complex Number: "<<endl;

cout<<"Real Part: ";

cin>>real;

cout<<"Imagenary Part: ";

cin>>imag;

complex c2(real,imag),c3;

cout<<"Sum of Two Complex Numbers: "<<endl;

c3.sum(c1,c2);

c3.show();

cout<<endl;

cout<<"Subtraction of Two Complex Numbers: "<<endl;

c3.sub(c1,c2);

c3.show();

cout<<endl;

cout<<"Multiplication of Two Complex Numbers: "<<endl;

c3.multiply(c1,c2);

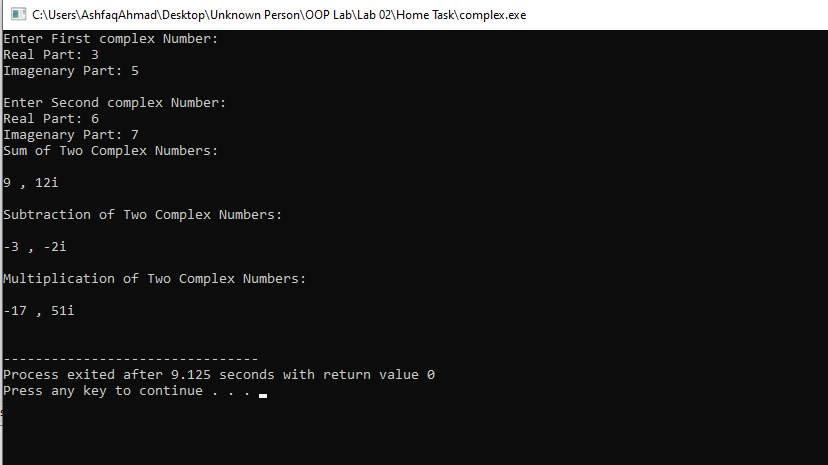
c3.show();

cout<<endl;

return 0;

};

**Compilation:**



**Task no 2:**

Reuse Complex class written in Activity 2.4.1 to modify the addCom(), subCom(), and mulCom() class functions. Instead of passing two objects in each, now pass only one object. Change the return type of each function to complex. Adjust the function code to match the changes. Demonstrate and test the modified class and its objects using test cases given in Section 2.5.

**Source code:**

#include <iostream>

using namespace std;

class complex

{

private:

double re,im;

public:

void input();

complex addcom(complex c1);

complex subcom(complex c1);

complex mulcom(complex c1);

void display();

};

void complex::input()

{

cout<<"enter real part: ";

cin>>re;

cout<<"enter img part: ";

cin>>im;

}

complex

complex::addcom(complex c1)

{

complex temp1;

temp1.re=c1.re+re;

temp1.im=c1.im+im;

return temp1;

}

complex

complex::subcom(complex c1)

{

complex temp1;

temp1.re=re-c1.re;

temp1.im=im-c1.im;

return temp1;

}

complex

complex::mulcom(complex c1)

{

complex temp1;

temp1.re=(c1.re\*re)-(c1.im\*im);

temp1.im=(c1.re\*im)+(c1.im\*re);

return temp1;

}

void complex::display()

{

if(im>0)

cout<<re<<"+"<<im<<"i"<<endl;

else

cout<<re<<im<<endl;

}

int main ()

{

complex c1,c2,c;

c1.input();

cout<<"\nc1 :";

c1.display() ;

c2.input();

cout<<"\nc2 :";

c2.display();

cout<<"\nsum :";

c=c2.addcom(c1);

c.display();

cout<<"\nDifference :";

c=c2.subcom(c1);

c.display();

cout<<"\nproduct :";

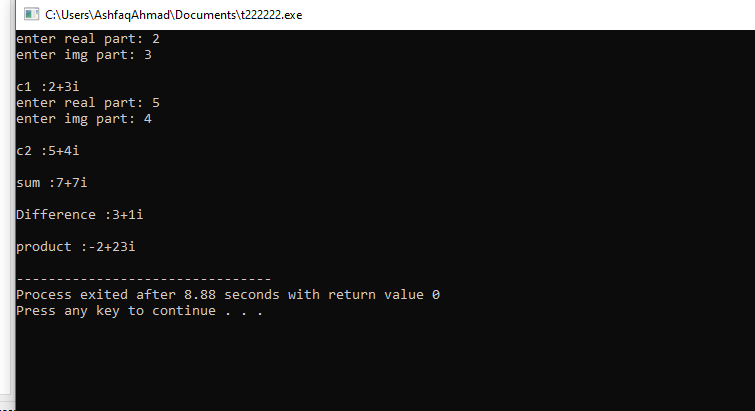
c=c2.mulcom(c1);

c.display();

return 0;

};

**Compilation:**



**Task no 3:**

Create a class called IntegerSet. Each object of class IntegerSet can hold integers in the range 0 through 49. A set is represented internally as an array of ones and zeros. Array element a[ i ] is 1 if integer i is in the set. Array element a[ j ] is 0 if integer j is not in the set. The default constructor initializes a set to empty set i.e., a set whose array representation contains all zeros. Provide member functions for the common set operations. For example,

1. Provide a newIntegerSet member function that initialize array to user‐defined array provided as an input in array notation. Note function return type is void.

2. Provide a unionOfIntegerSets member function that creates a third set which is the set‐ theoretic union of two existing sets (i.e., an element of the third set’s array is set to 1 if that element is 1 in either or both of the existing sets, and an element of the third set’s array is set to0 if that element is 0 in each of the existing sets).

3. Provide an intersectionOfIntegerSets member function that creates a third set which is the set‐ theoretic intersection of two existing sets (i.e., an element of the third set’s array is set to 0 if that element is 0 in either or both of the existing sets, and an element of the third set’s array is set to 1 if that element is 1 in each of the existing sets).

4. Provide an insertElement member function that inserts a new integer k into a set (by setting a[k] to 1).

5. Provide a deleteElement member function that deletes integer m (by setting a[m] to 0).

6. Provide a setPrint member function that prints a set as a list of numbers separated by spaces. Print only those elements that are present in the set (i.e., their position in the array has a value of 1).

7. Provide an isEqualTo member function that determines if two sets are equal. Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.

**Source code:**

#include<iostream>

using namespace std;

class IntegerSet

{

private:

bool Integer[50];

public:

IntegerSet();

IntegerSet unionOfIntegerSets(IntegerSet a,IntegerSet b);

IntegerSet intersectionOfIntegerSets(IntegerSet a,IntegerSet b);

bool isEqualTo(IntegerSet a,IntegerSet b);

void insertElement(int x);

void deleteElement(int y);

void showset();

};

IntegerSet::IntegerSet()

{

for(int i=0;i<50;i++)

{

Integer[i]=0;

}

}

void IntegerSet::insertElement(int x)

{

if(x>=0 && x<50)

{

Integer[x]=1;

}

}

IntegerSet IntegerSet::unionOfIntegerSets(IntegerSet a,IntegerSet b)

{

IntegerSet third\_set;

for(int i=0;i<50;i++)

{

if(a.Integer[i]==1 || b.Integer[i]==1)

{

third\_set.Integer[i]=1;

}

}

return third\_set;

}

IntegerSet IntegerSet::intersectionOfIntegerSets(IntegerSet a,IntegerSet b)

{

IntegerSet third\_set;

for(int i=0;i<50;i++)

{

if(a.Integer[i]==1 && b.Integer[i]==1)

third\_set.Integer[i]=1;

}

return third\_set;

}

void IntegerSet::deleteElement(int y)

{

if(y>=0 && y<50)

{

Integer[y]=0;

}

}

void IntegerSet::showset()

{

for(int i=0;i<50;i++)

{

if(Integer[i]==1)

{

cout<<" "<<i;

}

}

cout<<endl;

}

bool IntegerSet::isEqualTo(IntegerSet a,IntegerSet b)

{

for(int i=0;i<50;i++)

{

if(a.Integer[i]!=b.Integer[i])

return false;

else

return true;

}

}

int main()

{

cout<<"Enter the Set element to finish enter -1: "<<endl;

int element;

IntegerSet a , b , c ,d;

while(cin>>element,element!=-1)

{

a.insertElement(element);

}

cout<<"Set A: ";

a.showset();

cout<<endl<<"Enter Second Set element: "<<endl;

element=0;

while(cin>>element,element!=-1)

{

b.insertElement(element);

}

cout<<"Set B: ";

b.showset();

cout<<endl<<"The Union of set A and set B is U: ";

c=c.unionOfIntegerSets(a,b);

c.showset();

cout<<endl<<"The Intersection of set A and set B is I: ";

d=d.intersectionOfIntegerSets(a,b);

d.showset();

if(c.isEqualTo(a,b))

cout<<endl<<"Set A and Set B are Equal: "<<endl;

else

cout<<endl<<"Set A and Set B are not Equal: "<<endl;

cout<<endl<<"IF you want to delet some element from Set A then enter it: "<<endl;

element=0;

while(cin>>element,element!=-1)

{

a.deleteElement(element);

}

cout<<"After deleting some element from Set A: ";

a.showset();

cout<<endl<<"IF you want to delet some element from Set B then enter it: "<<endl;

element=0;

while(cin>>element,element!=-1)

{

b.deleteElement(element);

}

cout<<"After deleting some element from Set B: ";

b.showset();

cout<<endl<<"The Union of set A and set B is U: ";

c=c.unionOfIntegerSets(a,b);

c.showset();

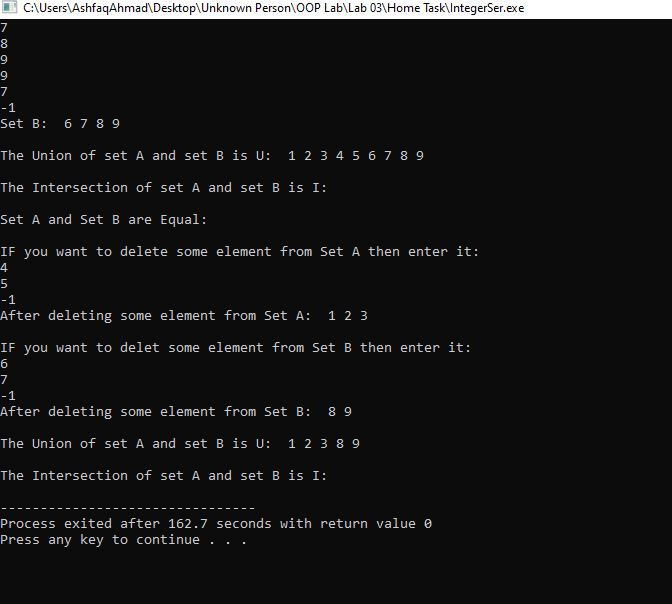
cout<<endl<<"The Intersection of set A and set B is I: ";

d=d.intersectionOfIntegerSets(a,b);

d.showset();

};

**Compilation:**



The end